WSI - ćwiczenie 5.
Sztuczne sieci neuronowe.

Adam Sudoł, 310933
Piotr Kosmala, 310754

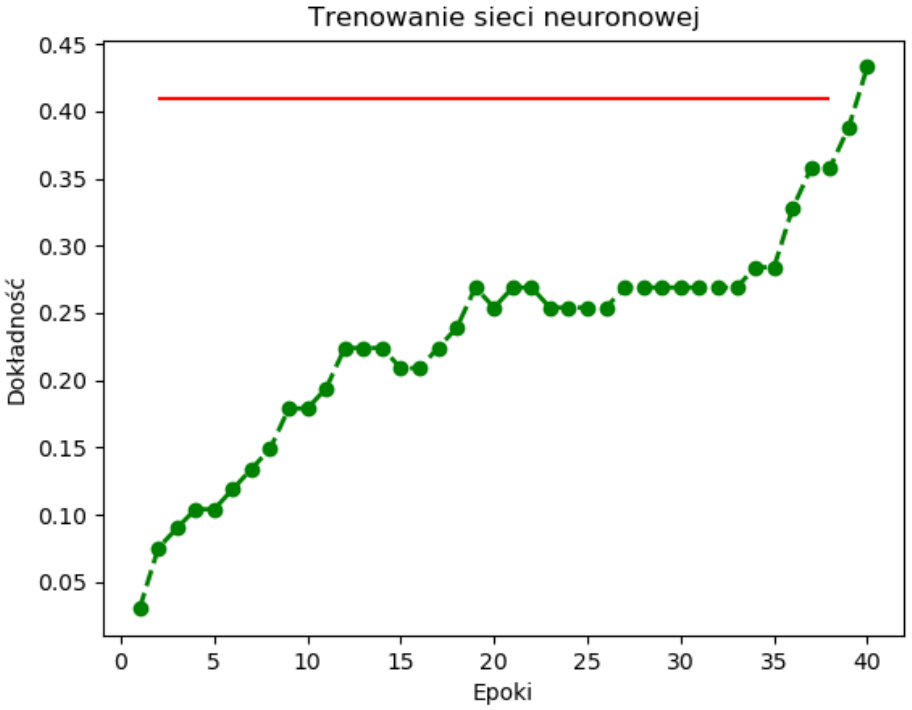
1. Cel laboratorium
   Celem tego ćwiczenia była implementacja perceptronu wielowarstwowego oraz wybranego algorytmu optymalizacji gradientowej z algorytmem propagacji wstecznej.
   Do optymalizacji gradientowej wybraliśmy metodę najszybszego spadku.

2. Opis trudności
   Najtrudniejsze, jak zazwyczaj, było przeniesienie koncepcji do kodu. Pisząc dokładniej, ogromną trudnością zarówno w 'wizualizacji' tego sobie, jak i napisaniu w pythonie, było przejście z neuronów do macierzy wag, aktywacji etc. Nie pomagały również materiały wykładowe, które były raczej trudne do zrozumienia.
   Innym trudnym elementem był *backward pass* z racji na komplikacje matematyczne, które używając wolframa nie byłyby aż tak uciążliwe, jednak implementacyjnie, gdzie rzadko piszemy stricte matematyczne rzeczy, były wyzwaniem.
   Problemem, którego na szczęście udało się uniknąć, były duże czasy obliczeń. Jest to zasługa głównie dobrze przygotowanych (i stosunkowo niewielkich) danych w pakiecie *sklearn*.
   Rzeczą nieistotną, ale minimalnie utrudniającą, był brak funkcji sigmoidalnej oraz jej pochodnej w znanych nam bibliotekach.
   Do uzyskania podziału na trzy różne pakiety dwa razy użyliśmy funkcji *train_test_split*, co w zupełności spełniło swoje zadanie.

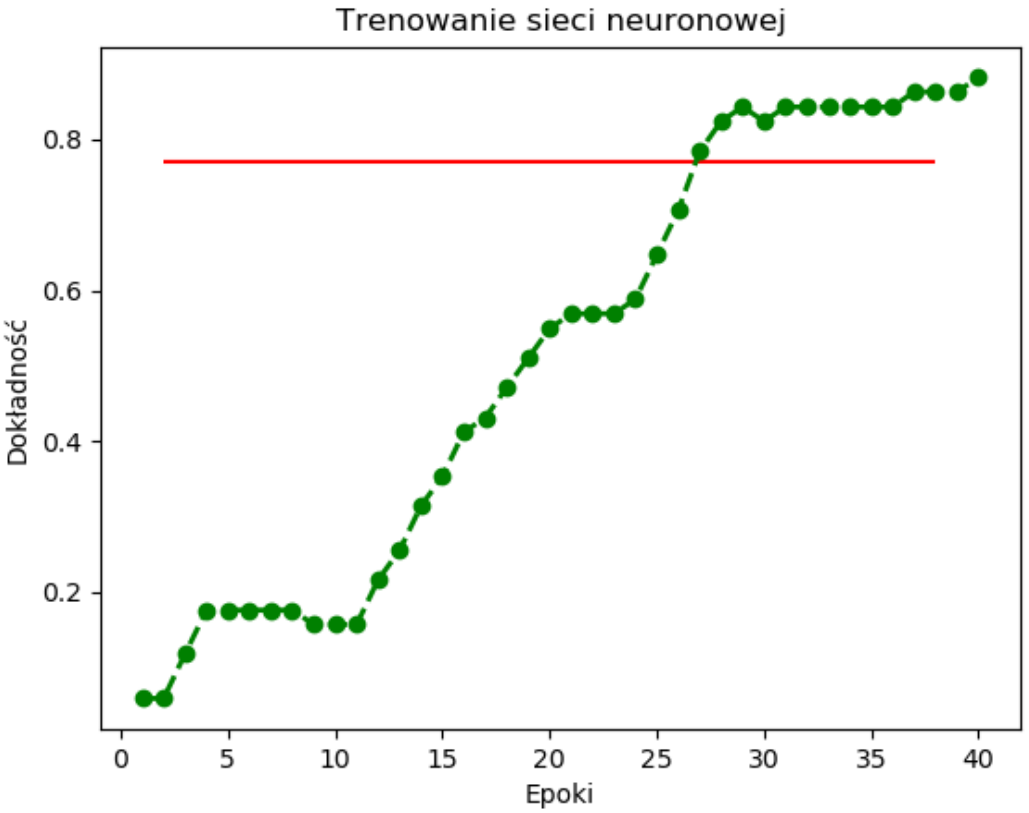
3. Eksperymenty

   Eksperymenty przeprowadzaliśmy w taki sposób, że manipulowaliśmy wielkością zestawu treningowego, walidacyjnego i testowego. Zmienialiśmy również liczbę neuronów w warstwach ukrytych. Przyjęliśmy jednak, że będziemy pracowali na dwóch warstwach ukrytych oraz czterdziestu epokach.
   Zapis 5:3:2 będzie oznaczał, że 0.5 z całego datasetu przeznaczono na część treningową, 0.3 na walidacyjną a 0.2 na testową.
   Czerwona linia oznacza skuteczność sprawdzoną na pakiecie testowym.
   Standardowo próbowaliśmy wybierać parametry skrajnie nieoptymalne, aby metodą prób i błędów dotrzeć do tych przystępnych. Przyjęliśmy już na początku, że w pierwszej ukrytej warstwie znajdą się 32 neutrony, w drugiej zaś 16, co sprawdziło się pozytywnie.

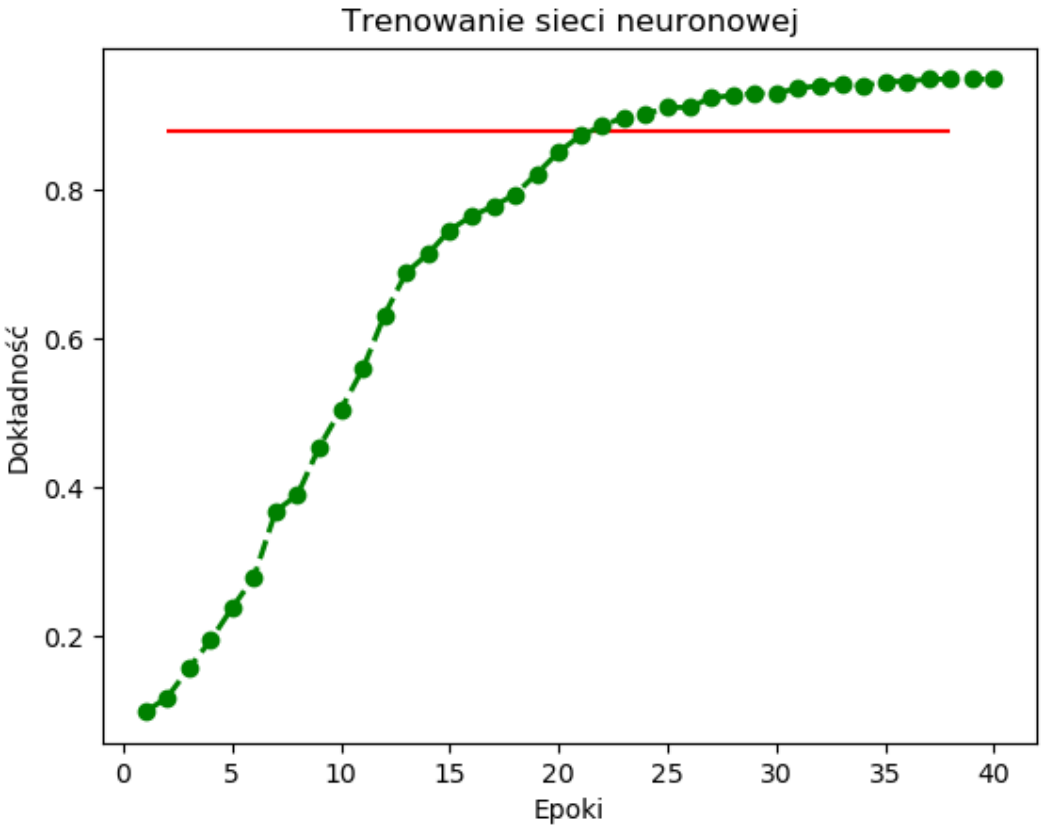
   W przypadku podziału danych 2:6:2 sieć działa źle. Na pakiecie walidacyjnym osiąga skuteczność maksymalnie 0.433, a na testowym 0.41. Nie jest to jednak zaskoczeniem z racji na mały pakiet treningowy.



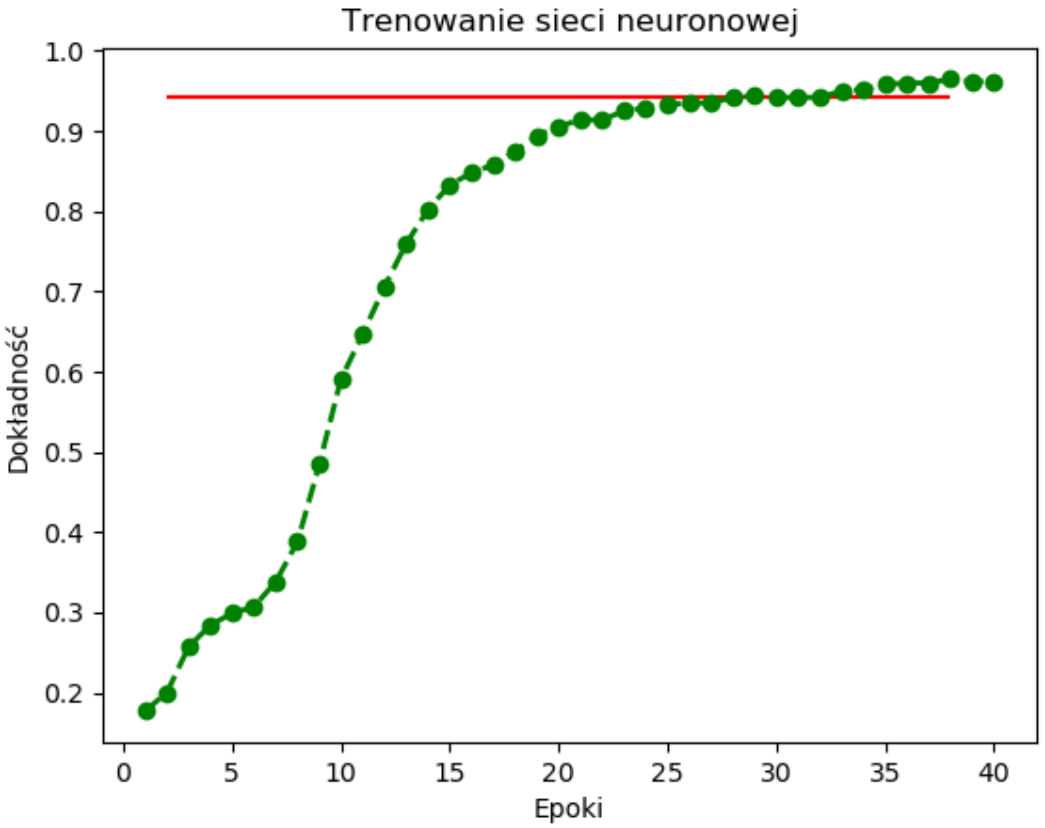
W przypadku podziału 4:2:4 działa lepiej, jednak jest to działanie pozornie znacznie lepsze. Sieć uczy się dosyć dobrze, ponieważ końcowo na pakiecie walidacyjnym osiąga skuteczność 0.882, jednak na pakiecie testowym skuteczność ta jest niższa i wynosi 0.77. Różnica ta wynika prawdopodobnie z tego, że zbiór testowy jest znacznie większy od zbioru walidacyjnego, przez co daje bardziej rzeczywiste wyniki.



W przypadku zestawu 8:1:1 sieć uczy się naprawdę dobrze. Końcowa dokładność wynosi 0.95, a dokładność na zbiorze testowym 0.88. Nawet oscylując wielkością zbioru testowego pomiędzy 0 a 2 wyniki są podobnie zadowalające.



Sieć uczy się najlepiej w przypadku 6:2:2. Dokładność końcowa wynosi 0.96, a dokładność testowa 0.94.



Powyższe eksperymenty robione były przy 32 i 16 neuronach w warstwach ukrytych. Dla podziału 6:2:2 zrobiliśmy również eksperymenty na 48 i 16 oraz 20 i 16 neuronach, jednak wyniki były podobnie zadowalające.